

Foliated Quantum Codes

A. Bolt,¹ G. Duclos-Cianci,² D. Poulin,² and T. M. Stace^{1,*}

¹*ARC Centre for Engineered Quantum System, Department of Physics,
University of Queensland, Brisbane, QLD 4072, Australia*

²*Département de Physique, Université de Sherbrooke, Québec, Canada
(Dated: July 12, 2016)*

We show how to construct a large class of quantum error correcting codes, known as CSS codes, from highly entangled cluster states. This becomes a primitive in a protocol that *foliates* a series of such cluster states into a much larger cluster state, implementing foliated quantum error correction. We exemplify this construction with several familiar quantum error correction codes, and propose a generic method for decoding foliated codes. We numerically evaluate the error-correction performance of a family of finite-rate CSS codes known as turbo codes, finding that it performs well over moderate depth foliations. Foliated codes have applications for quantum repeaters and fault-tolerant measurement-based quantum computation.

Quantum error correction is critical to building practical quantum information processors (QIP). In an influential series of papers, Raussendorf et al. described a measurement based approach to fault tolerant quantum processing using highly-entangled *cluster states*, defined on a 3D lattice [1–4]. Raussendorf’s 3D cluster state can be visualised as a *foliation* of Kitaev’s surface code [5, 6], i.e. a sequence of 2D surface code ‘sheets’, stacked together to form a 3D lattice. This is evident in [1], where it is shown that measuring the ‘bulk’ qubits of a 3D cluster state leaves the two logical surface-code qubits encoded in the boundary faces in an entangled Bell-pair.

Raussendorf’s 3D cluster gained prominence for its high fault-tolerant computational error thresholds $\lesssim 1\%$. It has applications in various QIP tasks, including long-range entanglement sharing, in which surface-code cluster states are created at regularly spaced local nodes, which are linked by medium-range optical channels into a 3D cluster state [7]. It is capable of fault-tolerant, measurement-based quantum computation, using an elegant geometric construction that braids defects in the interior of the 3D cluster state to produce robust Clifford gates. Universality is afforded by magic state injection and distillation [3, 4, 8].

The robustness of [3, 4] is inherited from the underlying surface code, which has a high error-correction threshold $\sim 11\%$ [6, 9–11]. The surface code has large distance, and zero-rate (the asymptotic ratio of the number of logical and physical qubits), reflecting the trade-off between distance and rate in two spatial dimensions [12]. It is natural to ask how to adapt the foliated structure of [1, 2] to use other underlying codes that could achieve a higher encoding rate.

Another motivation for our work is recent fault tolerant schemes that produce a universal gate set by code deformation and code switching [13–15]. Extending code foliation to codes that circumvent magic state distillation [8] may produce cluster states with lower resource overhead for fault-tolerant measurement-based QIP.

In this letter we show that all Calderbank-Steane-Shor

(CSS) codes can be *clusterized*, meaning that they can be derived, using single-qubit measurements, from a larger cluster state [16, 17] defined over the code qubits plus additional ancilla qubits. We use this fact to develop our main result: generalising Raussendorf’s 3D lattice to a foliation of any clusterized CSS code. This is a larger cluster state comprised of alternating copies of a clusterized CSS code and its dual. We demonstrate this construction for some familiar CSS codes, and present a general decoding algorithm for foliated codes, utilising the underlying code’s decoder. Finally, we apply the construction to a family of finite-rate CSS codes called turbo codes [18, 19], and present Monte Carlo simulations of the error-correction performance of foliated turbo codes. *Background:* CSS code stabiliser generators are classified into two sets: $S_Z \in \{I, Z\}^{\otimes n}$ and $S_X \in \{I, X\}^{\otimes n}$, where Z and X denote the Pauli matrices [20]. An $[[n, k, d]]$ CSS code satisfies $k = n - (|S_X| + |S_Z|)$. We write a stabiliser in S_Z as $Z_{\vec{b}} \equiv \otimes_{\vec{b}} Z^{b_j}$ for some binary vector $\vec{b} = (b_1, b_2, \dots, b_n)$ with $b_j = 1$ if qubit j is in the stabilizer $Z_{\vec{b}}$, and $b_j = 0$ otherwise, i.e. \vec{b} is a row of the code’s parity check matrix, B_Z . Similarly, a stabilizer in S_X is given by $X_{\vec{c}}$ for some binary list \vec{c} . The associated dual code is derived from the primal code by exchanging X and Z operators in the stabilizer generators, $X \leftrightarrow Z$.

A *cluster state* is defined on a collection of qubits located at the vertices of a graph [16, 17, 21]. A qubit at vertex v is associated with a cluster stabiliser $C_v = X_v(\otimes_{\mathcal{N}_v} Z) \equiv X_v Z_{\mathcal{N}_v}$, acting on it and its neighbours, \mathcal{N}_v . The cluster state is the $+1$ eigenstate of the C_v ’s.

Clusterized CSS codes: An $[[n, k, d]]$ CSS code can be generated from a larger *progenitor* cluster state, i.e. *clusterized*. The progenitor cluster is simply the cluster state associated to the Tanner graph of S_Z [22], i.e. a bipartite graph $G = (V, E)$ whose vertices V are labeled by code qubits j , or ancilla qubits a , each associated to a stabilizer $Z_{\mathcal{N}_a} \in S_Z$, so that $|V| = n + |S_Z|$. E contains the graph edge (a, j) if $[B_Z]_{a,j} = 1$. We now show that a codestate of the CSS code is obtained by measuring the

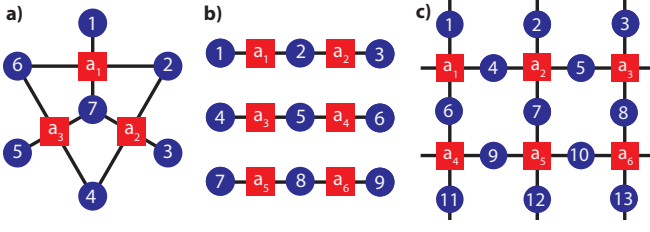


FIG. 1: Examples of progenitor clusters for clusterized CSS codes. **a)** Clusterized Steane code. **b)** Clusterized Shor code. **c)** Clusterized surface code. Code qubits (blue circles) are connected by cluster bonds (black lines) to ancilla qubits (red squares). An X -basis measurement of ancilla a_k projects neighbouring code qubits onto an eigenstate of $\otimes_{\mathcal{N}_{a_k}} Z \in \mathcal{S}_Z$.

ancilla qubits of the progenitor cluster in the X basis.

In the above definition, the cluster stabiliser associated to ancilla a is $C_a = X_a Z_{\mathcal{N}_a}$. Measurement of a in the X basis with outcome ± 1 projects adjacent code qubits into an eigenstate of the code stabiliser $Z_{\mathcal{N}_a} \in \mathcal{S}_Z$. Thus, \mathcal{S}_Z is generated by ancilla measurements.

Because \mathcal{S}_X and \mathcal{S}_Z mutually commute, the progenitor cluster is also an eigenstate of the generators in \mathcal{S}_X . To see this, take an element $X_{\vec{c}} \in \mathcal{S}_X$, and consider the product of cluster stabilizers centred at each code qubit $c_j \in \vec{c}$, given by $C_{\vec{c}} \equiv \otimes_{\vec{c}} C_{c_j} = \otimes_{\vec{c}} (X_{c_j} Z_{\mathcal{N}_{c_j}})$. The neighbourhood, \mathcal{N}_{c_j} , of code qubit c_j consists only of ancilla. The code stabiliser $X_{\vec{c}} = \otimes_{\vec{c}} X_{c_j}$ has even overlap with any Z -like stabiliser, $Z_{\vec{b}_a}$ (which is generated by measurement of ancilla a in the X -basis). It follows that the intersection of \vec{c} and \vec{b}_a has an even number of qubits. Any ancilla qubit a thus appears in the product $\otimes_{\vec{c}} C_{c_j}$ an even number of times, so $\otimes_{a \in \mathcal{N}_{c_j}} Z_a = \otimes_{a \in \mathcal{N}_{c_j}} \mathbb{I}_a$, and $C_{\vec{c}} = X_{\vec{c}}$. Thus, \mathcal{S}_X is generated by cluster stabilizers.

The same argument implies that logical X operators of the CSS code (which are products of local X operators that commute with \mathcal{S}_Z), are also generated by cluster stabilizers. It follows that ancilla measurements project the cluster state into a logical X codestate.

The surface code [5] exemplifies the relationship between a CSS code and a progenitor cluster state. Starting from the cluster state defined on the lattice shown in Fig. 1c, and measuring the ancilla qubits (red squares) in the X basis results in a new state on the remaining code qubits (blue circles) which is stabilised by the surface-code plaquette operators, e.g. $Z_2 Z_4 Z_5 Z_7 \in \mathcal{S}_Z^{\text{surf}}$, and vertex operators, e.g. $X_4 X_6 X_7 X_9 \in \mathcal{S}_X^{\text{surf}}$. It is therefore a codestate of the surface code [41].

Other examples of clusterized CSS codes are shown in Fig. 1a for Steane's 7-qubit code [23] for which $\mathcal{S}_Z^{\text{Steane}} = \{Z_1 Z_2 Z_6 Z_7, Z_2 Z_3 Z_4 Z_7, Z_4 Z_5 Z_6 Z_7\}$ (which is also a minimal example of the colour code [24, 25]); and Fig. 1b for Shor's 9-qubit code [26, 27] for which $\mathcal{S}_Z^{\text{Shor}} = \{Z_1 Z_2, Z_2 Z_3, Z_4 Z_5, Z_5 Z_6, Z_7 Z_8, Z_8 Z_9\}$.

The examples in Fig. 1 illustrate the fact that X -measurements of the ancilla qubits project out code stabilizers in \mathcal{S}_Z , while each stabiliser in \mathcal{S}_X comes 'for free' simply by considering products of C_j 's acting on the corresponding code qubits, and noting that these products act trivially on the ancilla qubits. For example, in the Steane code cluster of Fig. 1a, it is straightforward to check that $C_1 C_2 C_6 C_7 = X_1 X_2 X_6 X_7 \in \mathcal{S}_X^{\text{Steane}}$.

Foliated codes: Raussendorf's 3D cluster state construction [1–4], Fig. 2c, can be viewed as a *foliation* of the surface code cluster state shown in Fig. 1c. Alternating 'sheets' of the primal surface code cluster state and its dual are stacked together [42], with additional cluster bonds (green lines) extending between code qubits in each sheet and the corresponding code qubits in the adjacent dual sheets.

We now generalise this construction to arbitrary CSS codes. Take an alternating stack of sheets of clusterized primal and dual codes, and link the sheets together by creating additional cluster bonds between primal code qubits in a given sheet, m , and the corresponding dual code qubits in the adjacent sheets, $m \pm 1$. We call this a *foliated code*. The number of *layers*, L , in the foliated construction counts the number of primal–dual sheet pairs, so that $1 \leq m \leq 2L + 1$.

Fig. 2a shows the example of a foliated Steane code, (which is self-dual, so the primal and dual sheets are identical). Fig. 2b shows the foliated Shor code, for which primal and dual clusters are different. One can readily verify that this definition preserves the key feature of Raussendorf's construction: measuring the bulk qubits and the boundary ancilla qubits leaves the two boundary sheets in an encoded Bell-state [1]. Because this feature enables fault-tolerant measurement-based quantum computation and long-range entanglement sharing, our generalization has immediate applications in these settings, offering additional flexibility in the choice of code.

Errors: Errors may arise during construction of the cluster, storage of the qubits, or during single-qubit measurement. In the error models we consider, preparation and measurement errors can be mapped to possibly correlated storage errors [9]. Furthermore, after the cluster is created, since we perform single qubit X measurements, X errors do not affect the measurement outcome; only Z errors on the final foliated cluster act nontrivially. We note that X errors during cluster construction are equivalent to correlated Z errors in the final cluster [9, 10]. This asymmetry between X and Z errors is a consequence of the asymmetry in the definition of the cluster stabilizers. Correlated or asymmetric errors may also arise in specific applications, such as long-range repeaters where inter-node quantum transmission errors are much worse than those within a node, which can be mitigated by suitable choice of code [28].

Parity check operators: Errors in the foliated cluster are detected by parity check operators: a Z error will

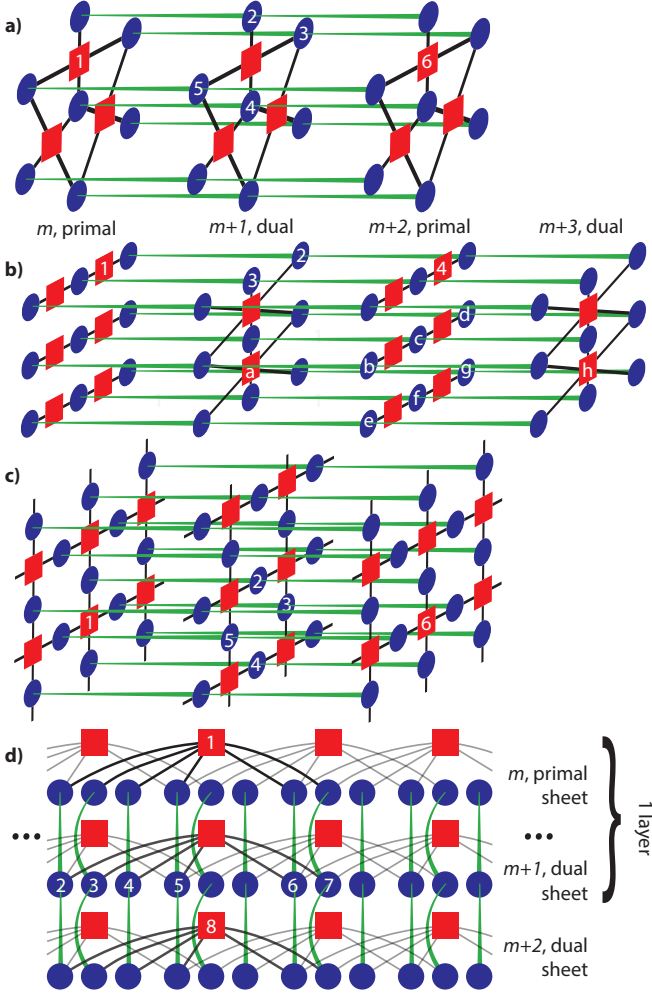


FIG. 2: Examples of foliated, clusterized CSS codes. Code qubits (blue circles) share cluster bonds (black lines) with ancilla qubits (red squares) in the same sheet, and with code qubits in adjacent sheets (green lines). **a)** Foliated Steane code. Being self-dual, primal and dual sheets are identical. The product of cluster stabilizers centred on the numbered qubits generates parity check operators $C_1 C_2 \dots C_6 = X_1 X_2 \dots X_6$. **b)** Foliated Shor code. This code is not self-dual, so primal and dual sheets are different, and there are two kinds of parity check operators, $C_a C_b \dots C_h = X_a X_b \dots X_h$ centred on primal sheets, and $C_1 C_2 C_3 C_4 = X_1 X_2 X_3 X_4$ centred on dual sheets. **c)** Foliated surface code [1], with parity check operators $C_1 \dots C_6 = X_1 \dots X_6$. **d)** Foliated self-dual convolutional code with parity check operator $C_1 \dots C_8 = X_1 \dots X_8$. Stabilizers are generated by translations of the kernel (indicated by thick edges) across frames (here, the frame length is 3).

flip one or more parity checks, giving a non-trivial error syndrome for the foliated cluster. Importantly, the parity check measurement outcomes can be inferred from sets of single-qubit X measurements.

Each parity check operator is associated with a CSS code stabiliser within a code sheet. To construct

a parity check operator, consider the CSS code stabiliser $X_{\vec{c},m} \in \mathcal{S}_X$, in sheet m of a foliated cluster state. The product of foliated cluster stabilizers centred on each of the code qubits indicated by \vec{c} is $C_{\vec{c},m} = X_{\vec{c},m} Z_{\mathcal{N}_{\vec{c},m}} = Z_{\vec{c},m-1} X_{\vec{c},m} Z_{\vec{c},m+1}$. The dual code sheets, $m \pm 1$, each have a cluster stabilizer $C_{a_{\vec{c}},m \pm 1} = X_{a_{\vec{c}},m \pm 1} Z_{\vec{c},m \pm 1}$ centred on an ancilla qubit $a_{\vec{c}}$ associated to \vec{c} . Thus, $\hat{P}_{\vec{c},m} \equiv C_{a_{\vec{c}},m-1} C_{\vec{c},m} C_{a_{\vec{c}},m+1} = X_{a_{\vec{c}},m-1} X_{\vec{c},m} X_{a_{\vec{c}},m+1}$ defines a parity check for the foliated cluster, centred on code stabilizer $X_{\vec{c},m}$. Note that parity check operators centred on primal sheets share no common qubits with those centred on dual sheets.

This generalises the construction of the parity check operators for Raussendorf's 3D cubic lattice, which are formed by products of X operators on the faces of the cubic unit cells, as shown in Fig. 2c (exemplified by numbered qubits). Parity check operators for other foliated CSS codes are exemplified by labelled qubits in other panels of Fig. 2. In a non-self-dual code, such as the Shor code, primal and dual parity check operators may have different weights, Fig. 2b.

Logical code operators within a sheet commute with the parity check operators. It follows that for an underlying $[[n, k, d]]$ code, there are weight- d undetected error chains on the foliated cluster, as in [1–4, 6]. Since the structure of the code in the direction of foliation is a simple repetition, it follows that the foliated cluster inherits the distance of the underlying code.

Decoding: A non-trivial error syndrome indicates the presence of Z errors. If the error probability is sufficiently small, the most likely class of errors can be inferred from the syndrome with high probability, facilitating error recovery. Small codes can be decoded by brute-force, but this is not computationally scalable in n .

There are a number of computationally efficient, near-optimal decoders available for both the 2D surface code and its 3D foliation, including hard decoders (which return a specific high-likelihood error pattern) based on perfect matching [6, 9], and soft decoders (which return a probability distribution over error patterns) based on renormalisation methods [11, 29].

Surface code decoders naturally generalise to the 3D Raussendorf lattice, as exemplified by matching-based decoders. While generic CSS codes cannot typically be efficiently decoded, many exact or heuristic decoders are known for specific code constructions [6, 11, 19, 30]. The problem we address here is to use a soft decoder for the underlying CSS code – which we presume to be efficient – as a subroutine in a decoder for the foliated construction. We describe a heuristic method based on belief propagation (BP) that may work in many cases [31, 32]. We assume the existence of soft decoders for the underlying CSS primal and dual codes, which, given a physical error model, calculates the probability of a Pauli error σ on code qubit j , $P(\sigma_j | S_{\text{CSS}})$, conditioned on a syndrome, S_{CSS} , which may itself be unreliable.

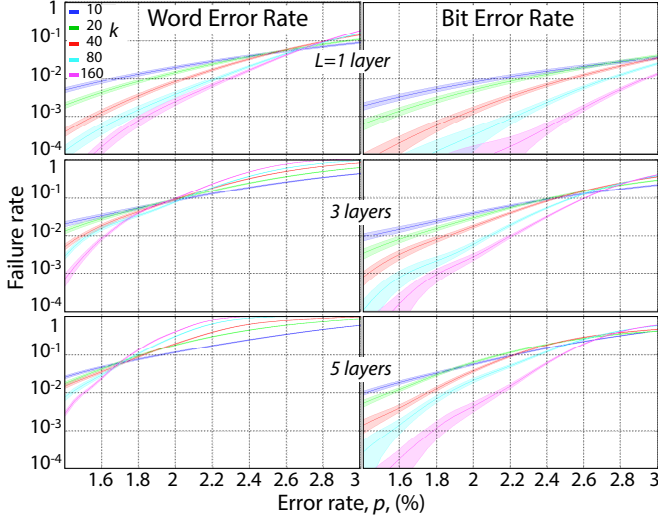


FIG. 3: Numerical performance results for a foliated $r = \frac{1}{25}$, $d = 25$ turbo code, for different numbers of foliated layers, L (rows). Different colours correspond to different code sizes, $k = nr$; shading indicates $\pm 1\sigma$. A ‘layer’ consists of a primal code sheet and a dual code sheet (see Fig. 2d). Word Error Rate (left column) counts any error(s) across all k logical qubits. Bit Error Rate (right column) counts the failure rate per logical qubit.

In the foliated case, consider a parity check operator $\hat{P}_{\vec{c},m} = X_{a_{\vec{c},m-1}}X_{\vec{c},m}X_{a_{\vec{c},m+1}}$, centred on primal sheet m . A non-trivial syndrome can arise because of errors on code qubits \vec{c} within code sheet m , or due to errors on the corresponding ancilla qubits, $a_{\vec{c}}$ in adjacent dual sheets $m \pm 1$. If the dual-sheet ancilla qubits were error-free, then all the parity check failures would be due solely to in-code qubit errors, so that the parity check outcomes centred on sheet m would be in direct correspondence with the CSS code syndrome for that sheet. The code syndrome could then be used in the CSS decoder to calculate a soft error model on sheet m , from which an error correction strategy could be determined.

However, errors on the dual-sheet ancilla qubits mean that the in-sheet syndrome passed to the CSS decoder is itself unreliable. To account for dual-sheet ancilla errors, we embed the CSS decoder in a BP routine, as follows.

Step 1: For each code qubit, j , in sheet m , the CSS decoder calculates an in-sheet error model probability distribution, $P_m(\sigma_j|S_m \cup P_{m \pm 1}(a_k))$, subject to both the measured code syndrome, S_m , which is derived from the foliated parity-check operators centred on sheet m , and an assumed error model, $P_{m \pm 1}(a_k)$, for errors on ancilla qubits, a_k , in adjacent dual sheets. **Step 2:** Using the result of Step 1 we fix the code qubit error model, $P_m(\sigma_j)$, and calculate an error model on the dual-sheet ancilla qubits, $P_{m \pm 1}(a_k|S_{m \pm 1} \cup P_m(\sigma_j))$. **Step 3:** We iterate Step 1, using the result of Step 2 for $P_{m \pm 1}(a_k)$, repeating until each error model converges.

Turbo Codes: We now consider the class of turbo codes, which are finite-rate CSS codes with bounded-weight stabilizers [19, 31, 33]. These are capable of encoding an arbitrary number of logical qubits with finite rate, $r = k/n$. Essentially, turbo codes are formed from a concatenation of two *convolutional* codes: an inner code G_I , and an outer code G_O [30, 34–36], each of which can be decoded with soft trellis decoders [19, 31, 33, 36, 37].

Convolutional codes are defined over an ordered set of qubits. The code stabilizers are generated by a kernel which is repeatedly translated across *frames* (i.e. blocks of the underlying physical qubits). For illustrative purposes, Fig. 2d shows a foliation of three sheets of a $d = 3$, $r = 1/3$, weight-6 self-dual CSS convolutional code cluster. The code stabiliser kernel is indicated by the dark cluster edges within a sheet. Turbo codes are conceptually similar, albeit with more complicated Tanner graphs.

Turbo codes provide a platform for testing the foliated construction on codes that are quite different to the surface code. Since they are a code family, we analyse the performance of the codes as a function of the code size $k = nr$, and the number of foliated layers, L . A soft trellis decoder [36, 38] for the underlying code is embedded as a subroutine in a BP decoder spanning the sheets of the foliation. The BP decoder run-time is linear in L , however the trellis decoder complexity is exponential in the size of the turbo code frame length, making simulations practically slow.

Fig. 3 shows the performance of a $d = 25$, $r = 1/25$, self-dual foliated turbo code, based on Monte Carlo simulations of errors. As noted earlier, X errors on the foliated cluster commute with parity check measurements. Thus, for our simulations we assume a phenomenological error model in which uncorrelated Z errors are distributed independently across the cluster with probability p . The decoder performance is quantified in terms of both word error rate (WER), which is the probability of one or more errors across all k encoded qubits, and the bit error rate (BER) which is the probability of an error in each of the encoded qubits.

For each L , there is a threshold error rate around $p \sim 2\%$, below which the code performance improves with code length (up to at least 160 encoded logical qubits per code sheet), consistent with pseudo-threshold behaviour seen in turbo codes [19]. As L increases, the threshold decreases, more pronouncedly for the WER than the BER. The range of k and L that we can simulate is limited by computational time, so we cannot explore the asymptotic performance for large L . Nevertheless, numerics indicate that foliated turbo codes perform quite well for moderate depth foliations.

We note that the foliated construction transforms a clusterised code into a fault tolerant resource state, but with a consequent reduction in threshold. This is seen in Fig. 3, and in Raussendorf’s construction in which the fault-tolerant threshold $\lesssim 1\%$ is smaller than the $\sim 11\%$

threshold for the surface code on which it is based. The $\sim 1\%$ threshold observed for foliated surface codes is obtained by scaling the code distance d and foliation depth L together. Here, the code distance is fixed at $d = 25$, which is responsible for the observed decreasing value of the pseudo-threshold with increasing L .

Our main motivation for studying turbo codes is to demonstrate the foliated construction and BP decoder in an extensible, finite-rate code family. Practically, these and other finite-rate codes may have applications in fault-tolerant quantum repeaters networks [7, 39], where local nodes create optimal clusterized codes to reduce resource overheads or error tolerance [40].

In conclusion, we have shown how to clusterize arbitrary CSS codes. We have shown how to foliate clusterized codes, generalising Raussendorf's 3D foliation of the surface code. We have described a generic approach to decoding errors that arise on the foliated cluster using an underlying soft decoder for the CSS code as a subroutine in a BP decoder, and applied it to error correction by means of a foliated turbo code. This construction may have applications where codes with finite rate are useful, such as long-range quantum repeater networks.

This work was funded by the ARC Centre of Excellence for Engineered Quantum Systems, NSERC and the Canadian Institute for Advanced Research. We thank Sean Barrett, Andrew Doherty, Terry Rudolph, Clemens Mueller and Stephen Bartlett for helpful discussions.

* stace@physics.uq.edu.au

- [1] R. Raussendorf, S. Bravyi, and J. Harrington, Phys. Rev. A **71**, 062313 (2005).
- [2] R. Raussendorf, J. Harrington, and K. Goyal, Ann. Phys. **321**, 2242 (2006).
- [3] R. Raussendorf and J. Harrington, Phys. Rev. Lett. **98**, 190504 (2007).
- [4] R. Raussendorf, J. Harrington, and K. Goyal, New J. Phys. **9**, 199 (2007).
- [5] A. Kitaev, Annals of Physics **303**, 2 (2003).
- [6] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, J. Math. Phys. **43**, 4452 (2002).
- [7] Y. Li, S. D. Barrett, T. M. Stace, and S. C. Benjamin, Phys. Rev. Lett. **105**, 250502 (2010).
- [8] S. Bravyi and A. Kitaev, Phys. Rev. A **71**, 022316 (2005).
- [9] T. M. Stace and S. D. Barrett, Phys. Rev. A **81**, 022317 (2010).
- [10] S. D. Barrett and T. M. Stace, Phys. Rev. Lett. **105**, 200502 (2010).
- [11] G. Duclos-Cianci and D. Poulin, Physical review letters **104**, 050504 (2010).
- [12] S. Bravyi, D. Poulin, and B. Terhal, Phys. Rev. Lett. **104**, 050503 (2010).
- [13] H. Bombin, arXiv:1412.5079 (2014).
- [14] A. Paetznick and B. W. Reichardt, Physical review letters **111**, 090505 (2013).
- [15] J. T. Anderson, G. Duclos-Cianci, and D. Poulin, Physical review letters **113**, 080501 (2014).
- [16] R. Raussendorf and H. J. Briegel, Phys. Rev. Lett. **86**, 5188 (2001).
- [17] R. Raussendorf, D. E. Browne, and H. J. Briegel, Phys. Rev. A **68**, 022312 (2003).
- [18] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, TDA Progress Report. 42-124. (1996).
- [19] D. Poulin, J.-P. Tillich, and H. Ollivier, Information Theory, IEEE Transactions on **55**, 2776 (2009).
- [20] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (CUP, 2000).
- [21] H. Briegel, D. Browne, Dür., R. Raussendorf, and V. d. Nest, Nature Physics **5**, 19 (2009).
- [22] R. Tanner, IEEE Transactions on Information Theory **27**, 533 (1981).
- [23] A. M. Steane, Phys. Rev. Lett. **77**, 793 (1996).
- [24] G. Duclos-Cianci, H. Bombin, and D. Poulin, American Physical Society, APS March Meeting (2011).
- [25] M. Kargarian, H. Bombin, and M. Martin-Delgado, New Journ. Phys. **12** (2010).
- [26] A. R. Calderbank and P. W. Shor, Phys. Rev. A **54**, 1098 (1996).
- [27] P. Shor, Phys. Rev. A **52-4**, R2493 (1995).
- [28] C. Cafaro and S. Mancini, Phys. Rev. A **82**, 012306 (2010).
- [29] G. Duclos-Cianci and D. Poulin, Quantum Information & Computation **14**, 721 (2014).
- [30] H. Ollivier and J.-P. Tillich, Phys. Rev. Lett. **91**, 177902 (2003).
- [31] R. McEliece, D. MacKay, and J. Cheng, IEEE Journal on Selected Areas in Communications **16**, 140 (1998).
- [32] D. Poulin and Y. Chung, Quantum Information & Computation **8**, 987 (2008).
- [33] S. Papaharalabos, Communications IET, 1751-8628. (2007).
- [34] H. Ollivier and J.-P. Tillich, INRIA, Project CODES, BP 105. (2009).
- [35] J.-P. Tillich and G. Zemor, in *Information Theory, IEEE International Symposium on* (2009).
- [36] P. Tan and J. Li, arXiv:1004.0174 (2010).
- [37] J. Geldmacher and J. Gotze, in *7th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)* (2012).
- [38] J. Tu, Zhenyu. Li and R. Blum, EURASIP Journal on Applied Signal Processing. **6**, 961-971 (2005).
- [39] L.-M. Duan, M. D. Lukin, J. I. Cirac, and P. Zoller, Nature **414**, 413 (2001).
- [40] T. Satoh, K. Ishizaki, S. Nagayama, and R. V. Meter, Phys. Rev. A **93**, 032302 (2016).
- [41] We allow ± 1 eigenstates of $Z_{\tilde{b}_a} \in \mathcal{S}_Z$ in this definition; a trivial syndrome corresponds to agreement between $Z_{\tilde{b}_a}$ and the associated ancilla X_a measurement outcome.
- [42] The dual to the surface code is also a surface code, albeit on the lattice geometrically rotated by 90 degrees.